Share

15 Java GUI

Created by GT F
Last updated 2019-08-14

グラフィカルユーザインタフェース（Graphical User Interface）。GUIでは、コンピュータの画面上に、ウィンドウ、アイコン、ボタンといったグラフィックが表示され、ユーザはそれらの中から目的の動作を表すグラフィックスをマウスなどのポインティングデバイスで選択する。

Java GUI 利用手順

1. ウィンドウを作る
2. レイヤーアウトと部品を追加する
3. イベントを処理する

具体的例

イベントを処理する。イベント発生を検知し、そのイベントに対応した処理を行うプログラムを書くには、**リスナーインタフェース**と呼ばれるインタフェースをインプリメントします。

ウィンドウを作る

トップcontainerクラス`JFrame`をインスタンスして、サイズを設定した後、表示させます。

```
1  import javax.swing.JFrame; // Java必要なPackageをインポートする
```

```
2
3  public class GUISample {
4      public static void main(String[] args) {
5          JFrame window = new JFrame("DCNet Java 教育"); // ウィンドウをインスタンス化する
6          window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); // 閉じるボタンをクリックすると、プログラミ
7          window.setSize(800, 600); // ウィンドウのサイズは 800 X 600 を設定する。
8          window.setVisible(true); // 画面が表示させる
9      }
10 }
```

レイヤーアウトと部品を追加する

ウィンドウにボタンを追加する

```
1  import javax.swing.JButton;
2  import javax.swing.JFrame;
3
4  public class GUISample {
5      public static void main(String[] args) {
6          JFrame window = new JFrame("DCNet Java 教育"); // ウィンドウをインスタンス化する
7          window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); // 閉じるボタンをクリックすると、プログラミ
8          window.setSize(800, 600); // ウィンドウのサイズは 800 X 600 を設定する。
9
10         JButton btn = new JButton("hello world"); // ボタンをインスタンスする
11         window.getContentPane().add(btn); // ウェンドウにボタンを追加する
12
13         window.setVisible(true); // 画面が表示させる
14     }
15 }
```

イベントを処理する

手順１：Listenerのクラスを新規作成する。

```
1  import java.awt.event.ActionEvent;
2  import java.awt.event.ActionListener;
3
4  public class ButtonClickListener implements ActionListener {
5      @Override
6      public void actionPerformed(ActionEvent e) {
7          System.out.println("ボタンクリックしました。");// コンソール：〝ボタンクリックしました。〟
8      }
9  }
```

手順２：Listenerを追加する。

```
1  import javax.swing.JButton;
2  import javax.swing.JFrame;
3
4  public class GUISample {
```

```
5
6       public static void main(String[] args) {
7           JFrame window = new JFrame("DCNet Java 教育"); // ウィンドウをインスタンス化する
8           window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); // 閉じるボタンをクリックすると、プログラミ
9           window.setSize(800, 600); // ウィンドウのサイズは 800 X 600 を設定する。
10
11          JButton btn = new JButton("hello world");
12          window.getContentPane().add(btn);
13
14          btn.addActionListener(new ButtonClickListener());
15
16          window.setVisible(true); // 画面が表示させる
17      }
18  }
```

代表なイベント

イベント名称	説明
ActionEvent	ボタンクリックする時
MouseEvent	Mouse移動／クリックする時
KeyEvent	キーボードを押下
WindowEvent	ウィンドウを閉じる、最大、最小する時

内部クラス

`ButtonClickListener`を内部クラスにする。

```
1   import java.awt.event.ActionEvent;
2   import java.awt.event.ActionListener;
3   import javax.swing.JButton;
4   import javax.swing.JFrame;
5
6   public class GUISample {
7       // 内部クラス
8       public static class ButtonClickListener implements ActionListener {
9           @Override
10          public void actionPerformed(ActionEvent e) {
11              System.out.println("ボタンクリックしました。");
12          }
13      }
14
15      public static void main(String[] args) {
16          JFrame window = new JFrame("DCNet Java 教育"); // ウィンドウをインスタンス化する
17          window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); // 閉じるボタンをクリックすると、プログラミ
```

```
        window.setSize(800, 600); // ウィンドウのサイズは 800 X 600 を設定する。

        JButton btn = new JButton("hello world");
        window.getContentPane().add(btn);

        btn.addActionListener(new ButtonClickListener());

        window.setVisible(true); // 画面が表示させる
    }
}
```

匿名クラス

`ButtonClickListener`を匿名クラスにする。

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;
import javax.swing.JFrame;

public class GUISample {

    public static void main(String[] args) {
        JFrame window = new JFrame("DCNet Java 教育"); // ウィンドウをインスタンス化する
        window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); // 閉じるボタンをクリックすると、プログラミ
        window.setSize(800, 600); // ウィンドウのサイズは 800 X 600 を設定する。

        JButton btn = new JButton("hello world");
        window.getContentPane().add(btn);

        // 匿名クラス
        ActionListener lsn = new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                System.out.println("ボタンクリックしました。");
            }
        };

        btn.addActionListener(lsn);

        window.setVisible(true); // 画面が表示させる
    }
}
```

ラムダ式

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;
```

```java
import javax.swing.JFrame;

public class GUISample {

    public static void main(String[] args) {
        JFrame window = new JFrame("DCNet Java 教育"); // ウィンドウをインスタンス化する
        window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); // 閉じるボタンをクリックすると、プログラミ
        window.setSize(800, 600); // ウィンドウのサイズは 800 X 600 を設定する。

        JButton btn = new JButton("hello world");
        window.getContentPane().add(btn);

        // ラムダ式
        ActionListener lsn = (ActionEvent e) -> {
            System.out.println("ボタンクリックしました。");
        };

        btn.addActionListener(lsn);

        window.setVisible(true); // 画面が表示させる
    }
}
```

代表のコンボネット

JLabel

ラベルは文字列またはImageを表示する可能です。

JTextField

ユーザー入力したテキストを取得し、設定します。

JCombox

ユーザー入力また選択可能のコンボネットです。

JList

ユーザーに複数選択肢を提供するコンボネットです。

JTable

データグリッドです。２次元データを表示、修正可能です。

FormEditor

従来のプログラマは手動でソースコードを書いて、画面を作成します。現在は様々IDEは可視化画面編集ツールを提供しているので、画面のビルドは簡単にできます。IDEのFormEditorを利用して、画面を作成します。

手順１：JFrameを新規作成します。

手順２：サンプルのJTableを追加します。

手順３：サンプルのJTableの細かいことを設定します。

手順４：ボタンJButtonを追加します。

手順５：イベントを実装する。

サンプルコード

サンプル１：JTableにレコードを全部クリアする。

```
public static void clear(JTable table) {
    DefaultTableModel model = (DefaultTableModel) table.getModel();
    model.setRowCount(0);
}
```

サンプル２：JTableにレコードを一行を追加する。

```
public static void append(JTable table, List<?> datas) {
    DefaultTableModel model = (DefaultTableModel) table.getModel();
    model.addRow(datas.toArray(new Object[0]));
}

public static void append(JTable table, String... datas) {
    DefaultTableModel model = (DefaultTableModel) table.getModel();
    model.addRow(datas);
}

public static void append(JTable table, Object... datas) {
    DefaultTableModel model = (DefaultTableModel) table.getModel();
    model.addRow(datas);
}
```

サンプル３：JTableにレコードをループする。

```
DefaultTableModel model = (DefaultTableModel) table.getModel();
for (int i = 0; i < model.getRowCount(); ++i) {
    Object[] vs = new Object[model.getColumnCount()];
    for (int y = 0; y < model.getColumnCount(); ++y) {
        vs[y] = model.getValueAt(i, y);
    }
    // TODO ここに実装してください。
}
```

サンプル４：JTableを動的に再ビルドする。

```
public static void resetTable(JTable jtable, String... columns) {
    DefaultTableModel dataModel = new DefaultTableModel();
    for (String column : columns) {
        dataModel.addColumn(column);
    }
    jtable.setModel(dataModel);
}
```

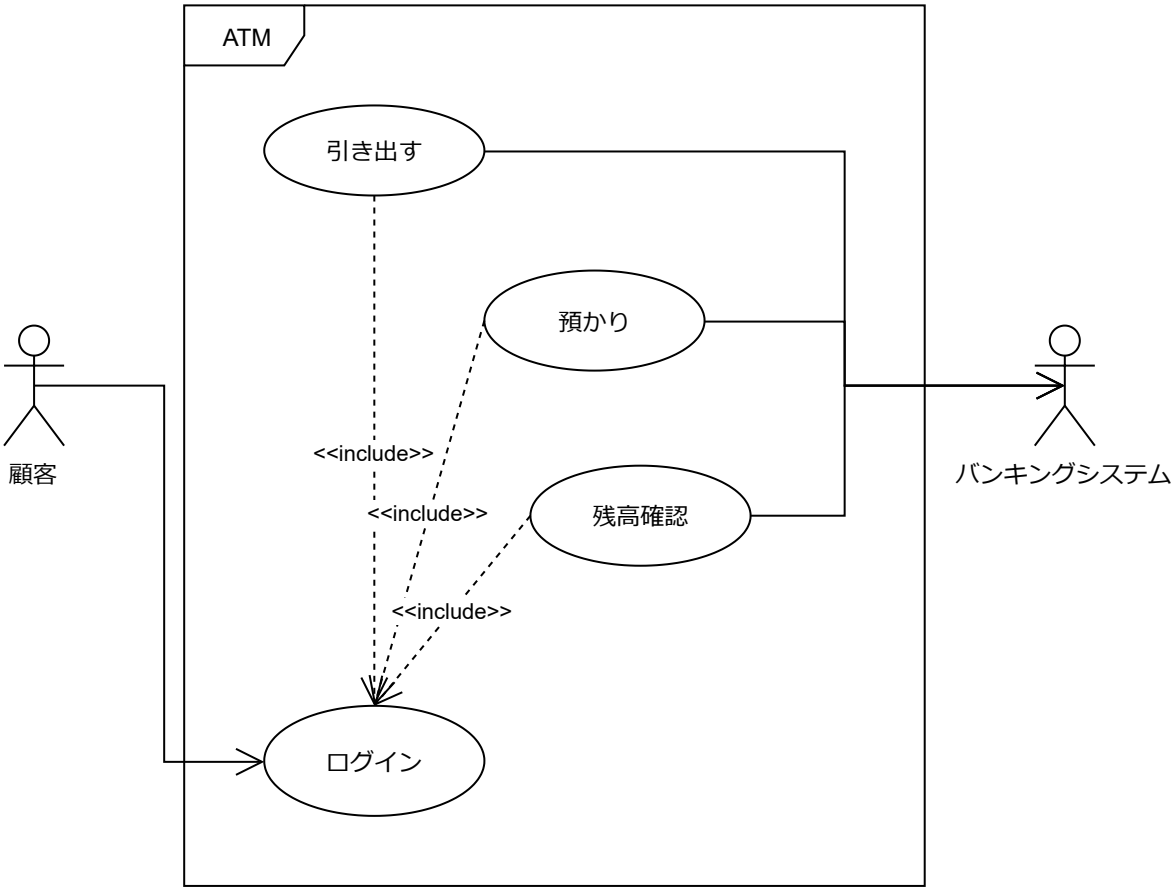
サンプル５：JTableを選択した行を取得する。

```
public static Object getSelectedRow(JTable table, int columnIndex) {
    int rowIndex = table.getSelectedRow();
    return table.getModel().getValueAt(table.
        convertRowIndexToModel(rowIndex), columnIndex);
}
```

実戦

以下ユースケース通り、ATMアプリを作成してください。

1. 顧客はキャッシュカードを持っている、キャッシュカードの番号は一意である。
2. バンキングシステムはCSVファイル顧客データを保存している。
3. 顧客が指定した金額でお金を引き出し、バンキングシステムから金額が引く。
4. 顧客が指定した金額でお金を預かり、バンキングシステムから金額が引く。
5. 顧客が指定した金額でお金を残高確認、バンキングシステム指定された顧客の残高を読み取り。



Like Be the first to like this

No labels